The Nonprofit Grant Operations Playbook

How mid-sized nonprofits run grant compliance, donor CRM, and restricted-fund reporting without a consulting project

By Angel Campa, founder of Ventora Labs

Published by GrantPipe - https://grantpipe.com

2026

The Nonprofit Grant Operations Playbook

How mid-sized nonprofits run grant compliance, donor CRM, and restricted-fund reporting without a consulting project

By Angel Campa, founder of Ventora Labs

Published by GrantPipe — https://grantpipe.com

How to use this playbook

This is written for the person inside a mid-sized nonprofit who is currently holding grant operations together with a spreadsheet, a folder, a CRM that isn't quite right, and a lot of memory.

It's not a vendor pitch. The last chapter talks about GrantPipe because that's what I build. Everything before that is framework, not product.

Read it linearly if you're early in the problem. Skip to whichever chapter matches what's on fire if you're not.

- **Chapter 1** — How the system you have now quietly stopped working.
- **Chapter 2** — What restricted-fund tracking actually requires.
- **Chapter 3** — The grant lifecycle from a compliance point of view.
- **Chapter 4** — Why most donor CRMs don't solve the grant problem.
- **Chapter 5** — Evaluating software without getting trapped.
- **Chapter 6** — Migrating off spreadsheets without losing six months.
- **Chapter 7** — A boring close process that actually holds.
- **Chapter 8** — What GrantPipe is, and who it's not for.

Chapter 1 — The system you have now

A mid-sized nonprofit almost never chooses its operating system. The system chooses itself.

Someone starts a spreadsheet because the grants list is getting hard to hold in one person's head. It gets shared. Columns get added. A copy gets emailed to the board chair before the quarterly meeting. Finance pulls numbers from it at audit time. The development director opens it before renewal conversations. A new grants coordinator joins and inherits it without a manual.

Three years later, that file is infrastructure. Nobody decided it would be. It just happened.

Meanwhile, the donor CRM came in for a different reason — somebody wanted cleaner acknowledgments, or a gift-entry workflow that wasn't four people typing the same donor's name differently. The CRM is probably good at what it does. It just doesn't know the grant exists.

Accounting is in QuickBooks. Maybe QuickBooks Online Advanced with classes, maybe Sage Intacct if the finance side grew up enough to budget for it. The transactions are clean. The problem is the transactions don't carry the grant context — which award they belong to, which restriction they touch, which budget line the funder cares about.

The evidence lives in a Google Drive folder. Receipts, award letters, activity reports, maybe some photos from programs. Organized by whoever organized it last. Findable if you know what you're looking for. Not findable if you don't.

The reporting deadlines live in a calendar, or a to-do list, or one person's head.

Five systems. One set of grant obligations. And between any two of those systems, the bridge is a person. Usually the same person.

That's what most mid-sized nonprofits actually run on. Not because anyone designed it that way. Because each piece arrived for a good reason and the stitching together was the kind of work nobody budgets for.

What stops working first

The stitching holds for a while. It stops holding in a predictable order.

First, version control goes. The spreadsheet gets edited in two places. Somebody saves a local copy. An email attachment becomes canonical for one person while the shared link becomes canonical for someone else. Nobody notices until a number doesn't match in a meeting.

Then, documentation drifts from the transactions. A program coordinator pays a vendor with a credit card. The receipt goes in the drive folder, eventually. The transaction goes to the ledger, immediately. The mental link between them lives with whoever approved the charge. That link is fine until that person is out for two weeks and a funder asks a question.

Then, restricted balances become interpretive. Finance has a balance. The grant manager has a balance. They're close. They're not the same. The reconciliation between them is a manual project that happens twice a year at most, usually during audit prep.

Then, reporting cycles start repeating the same work. The Q2 report to a funder requires the same assembly as Q1 did — pull the transactions, tie them to the right budget categories, collect the narrative, check that the numbers match what finance has, verify the evidence exists. Every quarter, from scratch.

Then, turnover becomes a crisis. When a grants manager leaves, the institutional memory that held the system together leaves with them. The next person spends two quarters reverse-engineering what columns mean and why certain conventions exist. During those two quarters, a funder sends a surprise question, and the organization doesn't have a clean answer.

None of these failures are dramatic. That's what makes them hard. They're slow leaks, not breaks. The organization keeps running. Reports still go out. Audits still happen. It just takes more hours than anyone tracks, and the hours are absorbed silently by a handful of senior people who shouldn't be spending their afternoons reconciling spreadsheets.

The cost you don't see on a budget

If you sat in on a finance meeting at a mid-sized nonprofit, you would not hear anyone say "we spent a lot on grant reconciliation this year." You would hear the CFO say the close took longer than expected. You would hear the development director say the federal report ate her whole week. You would hear the executive director say the audit was messy but manageable.

Those are the same cost. It just doesn't have a line item.

The cost is distributed across everyone's calendar in fragments. Twenty minutes here, forty there, a Sunday afternoon once a quarter. Senior people spending afternoons reconciling spreadsheets when nothing on their job description says that's the job. It never aggregates into a number the board sees, which is why it rarely gets fixed.

This is the baseline you're trying to reduce. Not to zero. Operations will always take some hours. But the gap between "we're running a system" and "we're chasing information inside a system" is real, and it's expensive.

Chapter 2 — What restricted-fund tracking actually requires

Most conversations about restricted funds treat the topic as an accounting problem. It is, partly. But the accounting is the easy part. The hard part is the operational layer underneath.

A restricted fund, from a compliance point of view, is a promise with a dollar value attached. The donor or funder said: you can use this money, but only for X, only during Y, only if Z happens. The organization agreed. The accounting system records the dollar amount. Everything else about the promise — the X, the Y, the Z — has to live somewhere.

The six facts every restricted balance needs

For any active restricted balance to be defensible — to auditors, to funders, to the board — the organization needs to be able to produce six facts on demand.

1. The source. Which donor, funder, or grant created the restriction. With the original letter, agreement, or pledge form attached. Not the paraphrase. The actual document.

2. The terms. What the money can be spent on, what it can't, what reporting is required, what the timeline is. In the funder's language, not the internal translation of it.

3. The opening balance. What was in the fund at the start of the current period — month, quarter, or fiscal year, whichever is relevant — and how that number reconciles with both the accounting system and the prior-period closing balance.

4. The activity. Every transaction that touched the fund this period, both revenue coming in and expenses going out. Each transaction tied to the specific restriction it satisfies, not just the general budget category.

5. The evidence. For every expense against the fund, the receipt, approval, or activity record that supports it. Findable, labeled, and tied to the transaction it supports — not floating in a folder somewhere.

6. The closing balance. What remains restricted. With documentation showing how the closing balance ties back to the opening balance plus activity.

If any of those six facts lives in someone's memory or requires reconstruction, the restricted balance is technically in the ledger but operationally unreliable. An auditor can find it. A board member can't verify it. A funder can't be given a clean answer without a half-day of preparation.

Where the gap usually opens

In practice, nonprofits don't fail at restricted-fund tracking in one dramatic way. They fail in the seams.

The ledger has the numbers. The grant file has the award letter. The operational tracker has the budget categories. The drive folder has the receipts. Four sources, four formats, and the reconciliation between them happens when somebody asks a question.

The test for whether your current setup is working isn't "can finance close the books." It's "can a staff member answer, in ten minutes, for any active restricted fund: what's the balance, how was it calculated, what evidence backs up the last three transactions, and what reporting is due next."

If that takes ten minutes, you have a system. If it takes an hour and three Slack messages, you have a collection of tools that happen to share an organization.

The accounting-versus-operations gap

One source of confusion worth naming: QuickBooks classes are not fund accounting.

It's a common setup. A nonprofit uses QuickBooks Online, turns on classes, and assigns a class per grant. This makes the P&L sortable by grant. It looks like fund accounting. Auditors accept it for smaller organizations.

What it doesn't do is model the restriction. A class is a tag on a transaction. It doesn't carry the award terms, the reporting deadlines, the funder's budget categories, or the evidence trail. It also doesn't prevent spending that exceeds the fund, or flag when activity is approaching a threshold.

The same applies to Sage Intacct at the lower configuration tiers, and to many QuickBooks Enterprise nonprofit setups. The accounting system is accounting for grants, which is not the same as managing them.

Real fund accounting, at the configuration level most mid-sized nonprofits need, is a software project with a consultant attached. Most organizations in the $500K to $10M range end up in a middle ground: accounting that tracks the money, operational systems that try to track the rest, and a reconciliation layer made of humans.

Chapter 3 — The grant lifecycle from a compliance point of view

Development teams think of the grant lifecycle as a pipeline. Prospect, research, cultivate, apply, steward. That framing is correct for the fundraising side.

From a compliance side, the lifecycle is different. It starts when the award arrives and keeps going long after the development team has moved on to the next ask.

The six stages of post-award compliance

Stage 1 — Intake. The award letter arrives. The organization has to capture the terms in a structured way: amount, period of performance, restrictions, reporting schedule, budget categories, any special conditions. This is the first place things go wrong. If the award letter gets filed without the terms being extracted into the operational system, every downstream stage inherits the gap.

Stage 2 — Setup. The grant becomes operational. Accounting sets up the class or fund code. Development notes the cultivation rhythm. Programs plans the activities. Budget categories get mapped to the funder's definitions. Reporting deadlines get calendared. Assignments get made.

Stage 3 — Execution. Activity happens. Money gets spent. Programs run. Reports get drafted. Evidence accumulates. During this stage, the tension between "record the work" and "do the work" decides whether the next stage is easy or miserable.

Stage 4 — Reporting. The funder's deadline arrives. The team assembles narrative, financial report, activity metrics, and evidence. If the prior stage captured everything in flight, this is an assembly job. If it didn't, this is an archaeology project.

Stage 5 — Closeout. The grant period ends. Remaining restricted balances get resolved. Final reports go in. Documentation gets archived. Lessons get captured, ideally.

Stage 6 — Audit support. Somewhere between three and twenty-four months after the grant ends, an auditor or monitor asks questions. If the documentation trail was built during execution, this is a retrieval job. If it wasn't, this is a panic.

Where the time actually goes

The pattern is predictable, even if the exact hours vary.

Setup is finite. A team sits down with the award letter, captures the terms, calendars the reports, and the stage is done.

Reporting is where teams feel pain. The cost comes from rebuilding information that wasn't captured in flight — chasing down receipts, reconstructing which program activity satisfied which budget category, retracing approvals from email threads. A report that would take an afternoon from a clean operating record takes several days when the information has to be reassembled.

Audit support has the same dynamic, amplified by time. Three to twenty-four months have passed. Staff have turned over. Documentation that felt findable last year has gotten harder to find. What could be a retrieval job becomes an archaeology project.

Stages 4 and 6 are where organizations feel the cost. Stage 3 — execution — is where the cost is actually generated. Every shortcut taken during execution becomes a bill that comes due in reporting and audit.

The "capture while it happens" principle

The single most useful habit in post-award compliance is this: capture the context at the moment of the transaction, not at the moment of the report.

When a program coordinator pays a vendor on a grant, the information needed for the funder report is available right then. Which grant. Which budget category. What activity this supports. Who approved it. Whether the receipt is attached.

If that information gets captured when the transaction happens, the report writes itself. If it gets captured months later when somebody sits down to write the report, the work is reconstruction, and reconstruction is always slower and less accurate than capture.

This is true across every stage, not just transactions. Capture the activity when it happens. Capture the milestone when it's hit. Capture the narrative beat when it's fresh.

Every system that makes this easy pays back the cost of the system. Every system that makes this harder — by requiring a separate login, a different spreadsheet, an email to the finance lead — costs the organization more than it saves.

Chapter 4 — Why most donor CRMs don't solve the grant problem

A mid-sized nonprofit that adopts a donor CRM — Bloomerang, DonorPerfect, Little Green Light, Keela, Neon, Givebutter — usually becomes happier at donor management and no better at grant management.

This isn't a criticism of those products. They do what they say. They track individual donors, manage gift records, run appeals, handle acknowledgments, and keep a relationship history. For a nonprofit whose revenue is mostly individual giving, any of them is a reasonable choice.

The gap opens for nonprofits whose revenue mix includes meaningful grant dollars. That's where the donor CRM's model stops matching the work.

Where the model breaks

A donor CRM thinks in terms of gifts from people or organizations. A grant doesn't fit that model cleanly.

A gift is a one-time event with an amount, a date, a donor, and a acknowledgment. A grant is a multi-stage obligation with a period of performance, a budget, reporting requirements, restrictions, evidence expectations, and a closeout process. You can record a grant as if it were a gift — many CRMs let you add a "Grant" record type — but the fields that matter for grant work aren't native to the product.

Specifically, most donor CRMs don't model:

- Multi-year awards with budget carryover between years
- Funder-defined budget categories that differ from internal categories
- Reporting deadlines with associated evidence requirements
- Restricted-fund balances that reconcile to the general ledger
- Budget-versus-actual tracking per budget category per period
- Subrecipient reporting for pass-through funds
- Indirect cost rates and their application to different categories

You can sometimes configure a workaround. A custom field here, a tagged transaction there. By the time you've configured all the workarounds, you've built a grant management system inside a donor CRM, and every future upgrade of the CRM threatens to break it.

The standard failure pattern

Here's what I see most often when a mid-sized nonprofit has adopted a donor CRM and is still struggling with grants.

The CRM has clean donor records. Individual giving is well-tracked. The development director loves the CRM and would not give it up.

Next to the CRM, there's a spreadsheet. It tracks the grants. It duplicates some information from the CRM — funder names, key dates — because the CRM isn't where people go for grant answers. The spreadsheet has its own conventions.

Next to both, there's a drive folder with award letters, agreements, and evidence.

Next to all three, there's QuickBooks with the transactions.

The development director works mostly in the CRM. The grants manager works mostly in the spreadsheet. The finance director works mostly in QuickBooks. The executive director works across all of them, usually by asking people for updates.

Each person's primary system is good at their job. The gaps between the systems are where every hard question lives.

What "one system" actually means

When I say GrantPipe puts donors, grants, restricted funds, and evidence on one operating record, I don't mean it replaces QuickBooks or absorbs the general ledger. The accounting system stays where it is.

What it replaces is the spreadsheet layer between the accounting system and the donor CRM. The layer where grant terms, reporting deadlines, budget-versus-actual, evidence, and the handoffs between finance and development currently live in people's heads and shared files.

Putting that layer into one system isn't about consolidation for its own sake. It's about eliminating the translation step. When finance and development look at the same grant record and see the same fields, they're not reconciling across systems. They're reading the same thing.

That's the wedge. Not a bigger CRM. Not a smaller ERP. A layer that exists because the current options are "spreadsheet" or "build it on Salesforce."

Chapter 5 — Evaluating software without getting trapped

Software evaluation at a mid-sized nonprofit tends to go off the rails in a predictable place. The demo is good. The sales engineer is sharp. The customer story sounds familiar. The energy shifts toward buying.

Then someone asks about implementation. And maintenance. And what happens when the consultant's engagement ends. And the answers get softer.

The only question that matters

Can our team operate this system after the buying process is over?

Everything else is a subquestion under that one. Not "does it do X." Not "is it enterprise-grade." Not "does it integrate with Y." Implementation dependence, consultant dependence, and configuration debt can turn a technically-capable system into something the team can't actually run.

Five subquestions worth writing down

1. What is the core job?

Be specific. "Better donor management" is too abstract. "Track 200 active donors, process monthly recurring gifts, pull LYBUNT reports, and tie donations to the right grant restrictions" is specific enough to test against.

If the core job is unclear going in, the biggest platform in the room always wins by being vague.

2. Who operates it after launch?

This is the question most evaluations skip. If the answer depends on a consultant permanently, put that consultant in the real budget, not the implementation budget. Salesforce NPSP is a useful example: the first ten licenses are donated through Power of Us, so the sticker price looks like zero — but the real budget item is implementation and ongoing configuration work, and that's the cost that actually determines whether the platform ships.

If the answer depends on one internal champion, plan for what happens when that person leaves.

If the answer is "the team, using the product as shipped," that's the strongest answer.

3. Where are the handoffs?

A good tool reduces handoffs. A tool that adds a new system the development lead checks while finance still uses accounting software and programs still use Drive has just added a handoff, not removed one.

Before buying, trace one full grant cycle through the proposed setup. Who does what, where, and what has to cross between them. If the handoff count went up, the tool is adding operational load.

4. Can the system explain its own numbers?

Storage isn't the same as proof. Anyone can store a document. Fewer systems can show why a restricted balance is $43,215 — which transactions rolled into it, which award letter defines the restriction, which budget categories constrain the spending.

If the staff can't defend the system's output without a spreadsheet, the system hasn't actually replaced the spreadsheet.

5. What's the exit risk?

If implementation stalls and the project fails, can the organization still understand its own data? Can the data be exported in a format another system can read? Has the organization taken on obligations it can't unwind?

This isn't pessimism. It's risk management. A platform where exit is painless is a platform that has to earn its keep every month.

Phrases worth slowing down on

A few things you'll hear during a demo that carry hidden costs.

- **"That can be customized."** Sometimes great. Sometimes it means the product doesn't match your workflow and configuration is going to be ongoing.
- **"Your consultant can configure that."** The consultant is a permanent line item.
- **"You can build a workflow for that."** Someone on your team becomes the workflow owner forever.
- **"That field can be repurposed."** You're about to use a marketing field to store compliance data.
- **"We can scope that in phase two."** Not in scope. Separate contract.

None of these are dealbreakers. All of them are worth writing down.

The one-hour evaluation

If you have one hour to evaluate a nonprofit CRM or grant management tool, spend it like this:

- 15 minutes on the product itself. Run through a real workflow you do every week.

- 15 minutes on the pricing page, including total cost of ownership.
- 15 minutes on reviews from organizations roughly your size and shape.
- 15 minutes on the data-export path. How would you leave if you had to.

Skip the demo for the first pass. Demos are optimized to make you want the product. Those four checks are optimized to make you understand it.

Chapter 6 — Migrating off spreadsheets

If you've decided the spreadsheet isn't the system anymore, the next question is how to move without losing three months to the migration.

The short version: don't migrate everything. Migrate the operating record for each active grant. Leave historical context where it is.

What actually needs to move

For every active grant, the new system needs:

- The award amount and terms
- The restrictions and funder rules
- The budget categories as the funder defined them
- The reporting cadence and what each report requires
- The current spend, tied back to ledger reality
- The evidence trail for what's been claimed so far
- The next action and who owns it

That's it. Seven fields per grant, plus attachments. A mid-sized nonprofit with five to ten active grants can migrate all of it in a week of focused work.

Historical grants — closed, audited, done — don't need to move. Keep the archive where it is. The new system is for active work.

What usually slows migrations down

Three things.

1. Trying to migrate donor history at the same time as grant operations. Donor history is big and messy. Grant operations are small and urgent. Do grants first, donors second, or in parallel with a different person owning each.

2. Configuration perfectionism. Every budget category mapped perfectly. Every custom field dialed in. Every workflow built. Skip it. Ship a boring configuration that handles 80% of cases, and refine the other 20% after a month of real use.

3. Doing the migration during a reporting month. Migrations during Q2 and Q4 close cycles collapse. Pick a slow month and move then.

A realistic week

If you give me a week and you have five active grants to migrate, here's what a realistic plan looks like.

Day 1: Set up the operating record for the first grant. Award letter attached, terms captured, budget categories defined, reporting calendar populated. This is the template for every subsequent grant. Do it carefully.

Day 2–3: Replicate the setup for the remaining four grants. Faster each time.

Day 4: Pull the last ninety days of transactions from QuickBooks and tag them to the right grant, restriction, and budget category. This is the reconciliation step. It's tedious but finite.

Day 5: Attach evidence for the last ninety days of transactions. Receipts, approvals, activity records. Whatever's missing gets chased down.

After that week, the operating record is real. The spreadsheet can become read-only. Future activity gets recorded in the new system as it happens, not in flight.

The trap to avoid

The biggest trap in a spreadsheet-to-software migration is keeping the spreadsheet running in parallel "just for now." Six months later, the parallel run hasn't stopped. Some staff look at the spreadsheet, others look at the software, and the reconciliation problem you were trying to solve has become a reconciliation problem between two systems you own.

Set a date. After that date, the spreadsheet is read-only reference. Future updates happen in the new system or they don't happen.

Chapter 7 — A close process that actually holds

Month-end close at a nonprofit is actually two closes.

The general ledger close asks whether the books tie out. That one's well-understood. Finance teams have been doing it forever.

The restricted-fund close asks whether the story around the money still makes sense. That one quietly eats days every month at most organizations.

The split-truth problem

Finance has the numbers. That's usually fine. The ledger is the ledger.

Development has the promise. They remember what the funder said the money was for. They have the award letter. They know which conversations happened.

Programs has the activity. They know what got done on the ground and whether it matched the workplan.

Leadership has the reporting burden. They're the ones who eventually have to put it in plain English for a board, a funder, or an auditor.

Four views. Four mental models. And very often, no single place where those four views get reconciled before month-end finishes.

The books close. The story doesn't.

Six questions a real close answers

For every active restricted balance or grant:

1. What balance was available at the start of the month?
2. What new restricted revenue came in?
3. What was spent against the fund?
4. What remains restricted now?
5. Does the documentation support every movement?
6. Are there any exceptions that need funder communication?

None of this is glamorous. That's the point. A good restricted-fund process should feel boring. No detective work on the balance. No search party for the receipts. No Sunday reconciliation.

Why the close fails when it fails

The usual reaction to a bad close is to blame the team. Almost always wrong. Nobody's being careless. The failure mode is that each department keeps a partial truth.

The work is translation. Finance translates transactions into budget-category language. Development translates funder language into internal language. Programs translates activity into reportable milestones. Leadership translates all of it into plain English.

Translation is where the time goes. And translation is invisible on the budget.

The five-step discipline

If you're fixing this without a software change:

1. **Reconcile each restricted balance.** Every fund. Every month. The balance matches the ledger, matches the award-letter obligation.

2. **Tie expenses back to the right restriction.** Not the general category. The specific grant or restriction the spending was approved against.

3. **Confirm what evidence exists.** Receipts, approvals, activity records. Whatever the funder will ask for.

4. **Flag anything that could affect funder reporting.** Missing documentation. Spending that exceeds a category. Activity that fell behind the workplan.

5. **Give finance and development the same final view.** A summary both sides sign off on before the month is officially closed.

Step five is what makes everything else hold. Without it, finance closes a month that development doesn't know is closed, and the next surprise arrives when a funder emails with a question.

Where software helps

Software doesn't fix the translation problem by itself. If four teams are using four systems, a fifth system won't make them agree.

But software can change what "agreement" looks like. When donor records, grant records, restricted funds, evidence, and reporting deadlines live on the same operating record, finance and development stop translating. They look at the same facts from different angles. The close stops being a reconciliation across systems and starts being a review inside one system.

That's what GrantPipe is trying to be. Not to replace the work. To remove the translation layer that eats hours every month.

Chapter 8 — What GrantPipe is, and who it's not for

GrantPipe is grant compliance and donor CRM software for mid-sized nonprofits. Donors, grants, restricted funds, and reporting evidence live on one operating record. Finance and development see the same answer to the same question. Report evidence accumulates while the work happens.

It's built for organizations with $500K to $10M budgets and three to ten active grants. It's priced at $99 per month for Starter, $249 for Growth, and $499 for Audit-Ready. One-month free trial, no credit card. Guided migration from Bloomerang, DonorPerfect, Salesforce, and spreadsheets.

Who it's not for

It's not for grantmaking foundations. Grantmakers have a different problem — managing outgoing awards, reviewing proposals, tracking grantees. Fluxx, Foundant, and SmartSimple own that space.

It's not for enterprise nonprofits with tens of millions in revenue and a dedicated Salesforce administrator on staff. Those organizations have usually justified Salesforce NPSP and should stay there.

It's not for organizations whose revenue is 95% individual giving and who don't have meaningful grant work. Bloomerang or DonorPerfect is probably the right answer.

It's for the organization in the middle. The one where grant work is real, the team is small, and the current setup is a spreadsheet plus a donor CRM plus a folder plus a lot of memory.

What a GrantPipe setup looks like

One record per grant. On it: award terms, budget, restrictions, reporting deadlines, spending-to-date, evidence, current status.

One record per donor. On it: gift history, preferences, relationship notes, restricted pledges, acknowledgments.

One reconciliation view that shows restricted balances across all grants, tied back to the accounting system.

One reporting calendar that surfaces what's due next, what evidence supports it, and who owns the draft.

Boring. That's the goal. Most of the work a mid-sized nonprofit does doesn't need excitement. It needs the obvious answers to be trustworthy when somebody asks.

The longer evaluation

If this is resonating and you want a longer evaluation guide, the one at https://grantpipe.com/resources/guides/how-to-choose-nonprofit-crm/ is what we send to organizations in active software decisions.

If the restricted-fund close is the part that hurts most, the checklist at https://grantpipe.com/resources/guides/restricted-fund-month-end-close-checklist/ is a working document teams use before the month ends.

If migrating from spreadsheets is where you are, the guide at https://grantpipe.com/resources/guides/migrating-from-spreadsheets-to-grant-management/ covers the week-by-week plan in more detail than this playbook has room for.

The product is at https://grantpipe.com. One-month free trial, no credit card.

About the author

Angel Campa is the founder of Ventora Labs, a Wyoming software studio focused on verticals enterprise SaaS usually ignores. Before GrantPipe he built CAMAudit, a commercial-lease audit product for corporate tenants. His current focus is GrantPipe, grant compliance and donor CRM software for mid-sized nonprofits that need cleaner records, funder reporting, and restricted-fund visibility without a consulting project.

- GrantPipe: https://grantpipe.com
- Ventora Labs: https://ventoralabs.com
- LinkedIn: https://www.linkedin.com/in/angelcampaventora/

This playbook is free to share. If it's useful to someone in your network, send it to them. If it's useful to you, the working version of most of this lives inside GrantPipe, and the trial is a month long with no credit card.